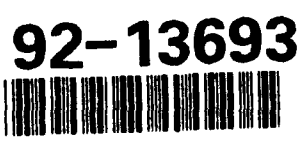
(2)

Mucit's Kyburgian Uncertain Inference Shell
User's Manual for version 1.00
Draft

Bülent Murtezaoğlu
mucit@cs.rochester.edu

August 15, 1990



REPORT DOCUMENTATION PAGE

Form Approved
OPM No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1990	Unknown

4. TITLE AND SUBTITLE

Mucit's Kyburgian Uncertain Inference Shell User's Manual for version 1.00

5. FUNDING NUMBERS

DAAB10-86-C-0567

6. AUTHOR(S)

Bulent Murtezaoglu

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Rochester
Rochester, NY 14627

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army CECOM Signals Warfare Directorate
Vint Hill Farms Station
Warrenton, VA 22186-5100

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

92-TRF-0023

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Statement A; Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

MKUIS is a research tool desinged for investigating various theoretical and practical issues concerning the statistical inference procedure developed by Henry Kyburg [1]. A subset of the procedure, limited to the homogeneous case, has been implemented in this version.

Since research about both the theoretical and the practical aspects of the systems is active and evolving, more powerful versions of this program will eventually become available.

14. SUBJECT TERMS

Artificial Intelligence, Data Fusion, Uncertain Inference

15. NUMBER OF PAGES

23

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Introduction

MKUIS is a research tool designed for investigating various theoretical and practical issues concerning the statistical inference procedure developed by Henry Kyburg [1]. A subset of the procedure, limited to the homogeneous case, has been implemented in this version.

Since research about both the theoretical and the practical aspects of the system is active and evolving, more powerful versions of this program will eventually become available.

Files

You should have the following files to use the program:
mukuis.lisp
front-end.lisp
math.lisp
io.lisp
definitions.lisp
sets.lisp
methods.lisp
genss.lisp
select.lisp
process.lisp
cover.lisp
xp.lisp
compclass.lisp

An additional file called *compile.lisp* can be used to compile all of the necessary program files.

The Syntax

A fully parenthesised prefix language has been implemented. Since the standard Common LISP function *read* is used as the input routine, minimal exposure to LISP would be sufficient for feeling comfortable with the syntax.

The input to the system is always a list. Comments starting with a ";" are allowed and disregarded both in the interactive mode and when input files are used.

Startup and Exit

A session is started by loading the file *mkuis.lisp* into the Common LISP interpreter. If for any reason the execution of the program is interrupted and

the control returns to the lisp top level, invoking the function *start-shell* should start the shell from scratch [1].

Typing the command *(bye)* at the shell prompt causes the program to exit.

Shell Variables

Shell variables are used to hold the values for various parameters needed by the inference engine. The values of shell variables and switches are set using the *set* command.

EG: (set confidence .9)

The shell variables used in this version are:

- **Confidence** The confidence level used to convert the sampling data to confidence intervals. Its default value is 0.95. Whenever a new confidence level is set, all the intervals corresponding to sampling data are re-computed.
- **Method** This is the variable to set to change the method used to compute the the reference class. Currently four values (1 to 4) are possible. A brief description of each method is given below. The report accompanying this manual should be consulted for detailed discussion of these methods. The default method is 4.
  - **Method 1** All the non-trivial candidate inference structures including all the XP style structures are found or constructed before looking for conflicts among them and selecting the reference class. This is the slowest of the methods and is intended for small (10-20 candidate reference classes) knowledge bases. For big sets of possible reference classes, it may be necessary to set the shell variable *xp-limit* to a value that arbitrarily stops the search to ensure reasonable execution times.
  - **Method 2** This is similar to method 1, but some attempt is made to decrease the number of inference structures before the XP's are constructed. An inference structure is excluded from the set of candidates before the XP's are constructed, if
    - it conflicts with and is dominated by another or
    - all the candidates it conflicts with are excluded by the first rule.
  - **Method 3** This is identical to method 3 except for an additional rule for exclusion:

[1] It is possible to crash the input routine and get dropped into the lisp debugger by putting commas etc. that *read* doesn't expect in you input. Restarting the shell might be necessary in those cases. Anything else that invokes the debugger is probably an unknown bug.

An inference structure is excluded from the set of candidates if there's a stronger (narrower interval) one that agrees with it.

This is the fastest method using XP's. It should be noted that, by limiting the precision of the endpoints of intervals, the search time can be bounded independent of the size of the set of possible inference structures.

- **Method 4** This is the fastest of the methods. The computation is identical to method 3 except instead of constructing the XP's, it constructs the interval cover of the remaining candidates after some are excluded by the above rules.

- **Precision** The number of decimal places taken as significant when testing for conflicts. It is set to 4 by default.

- **Trace** This flag is used to tell the program to print tracing information like lists of candidate reference classes and conflicts between them at various stages of computation. The permissible values are *on* and *off*. The default is *on*.

- **Time** When this flag is set, the program prints the timing information about the execution of queries. This information is acquired by a call the the LISP function time. The information printed may vary among LISP implementations. The default is *on*.

- **xp-limit** Limits the number of XP style inference structures that the program constructs. This limit depends on your LISP implementation and your patience. For small data files, all the possible XP's will be constructed in a reasonable amount of time, for large files this limit may be reached before XP generation is completed. The default is 62350.

Data Input Commands

Three kinds of data may be input:

1. Interval data of the form

$$(\%\,(< property >\, < set >)\,(p\ q))$$

Meaning that the proportion of the number of elements of the $< set >$ with $< property >$ is known to be in the interval $[p, q]$.

EG: (% (female human) (0.49 0.52))

Internally, % statements are interpreted as valid under any confidence level, so it is a good idea to give the program sampling data rather than probability data whenever possible. This shortcoming may be corrected in future versions of the program.

2. Sampling data of the form

$$(s\%\,(< property >< set >)\,(t\ s))$$

tell the program that out of t trials on members of $< set >$ s members have been discovered to have the $< property >$. Sampling data given by s% statements are converted to interval data by approximating binomial confidence intervals at the current level of confidence.

EG: (s% (male student) (4500 3000))

3. Subset relationship data of the form

$$(subs\ < set >\ (< set_1 >< set_2 > \ \ldots\ < set_n >))$$

Meaning that $< set_1 >, < set_2 >, \ldots, < set_n >$ are supersets of $< set >$.

EG: (subs cows (mammals animals))

The identical syntax is used for set membership statements:

$$(mem\ < object >\ (< set_1 >< set_2 > \ldots < set_n >))$$

EG: (mem tweety (bird penguin))

Notation for Sets

Any legal lisp symbol or number except the reserved symbols **I, XP, OR** may be used to denote sets. Set negation and unions are not permitted, set intersections are permitted. An intersection of sets is denoted by the list

$$(\mathbf{I} < set_1 >< set_2 > \ \ldots\ < set_n >)$$

Queries

The syntax for queries is:

$$(prob(< property >< object >))$$

to be read: "what is the probability of the $< object >$ having the $< property >$?."

EG: (prob (flyer tweety))

Miscellaneous Commands

Clear: clears the knowledge base. All the sampling, interval, subset and membership data are erased. This command is intended to enable the user to experiment with different sets of evidence within the same session.

Reset: completely resets the shell. Does a *clear* and sets the shell variables to their default values.

Load: loads an input file. The syntax is :

(load "file-name")

If tracing is on, the input is echoed to the standard output as it is read from the file. The prompt changes to *loading* ==> to avoid confusion.

Nesting of load commands is not allowed.

Suggestions

The code is written in standard Common LISP, so it can be compiled with no problems. Compiling the code is strongly recommended. If your compiler supports tail recursion elimination as an option, be sure to turn it on.

No session logging capability is built into the shell. For UNIX systems, I recommend running the program from within GNU Emacs, using the Inferior Lisp mode. The script utility of UNIX can also be used.

Reporting Bugs

All bug reports should be sent via e-mail to the author. Please describe the problem in detail and make sure that I can reproduce the erratic behaviour by sending a script of the session starting from the beginning.

Acknowledgements

I want to thank my advisor Henry Kyburg for his patience. The syntax of the language used in this implementation is heavily influenced by a similar program, CCRC, written by Ron Loui.

Sample Data File

```
;tweety first
(reset)
(set timing off)
(s% (fly bird) (100000 90000))
(s% (fly penguin) (1000 1))
(subs penguin (bird))
(mem tweety (bird))
(prob (fly tweety))
(mem tweety (penguin))
(prob (fly tweety))
(set confidence .99)  ;play with the confidence level
(prob (fly tweety))
(set confidence .51)
(prob (fly tweety))
(set trace off) ;trace demo
(prob (fly tweety))
(set timing on)         ;timing demo
(prob (fly tweety))

;interesting cases of dominance
(reset) ;clean start
(set timing off)
(% (p a) (.1 .2))
(% (p b) (.3 .4))
(% (p c) (.15 .45))
(subs c (a))
(mem x (a b c))
(prob (p x))
(clear)
(% (p a) (.1 .2))
(% (p b) (.3 .4))
(% (p c) (.35 .45))
(subs c (b))
(mem x ( a b c))
(prob (p x))
(clear)
(% (p a) (.1 .2))
(% (p b) (.3 .4))
(% (p (I a b)) (.35 .45))
(mem x (a b))
(prob (p x))
```

```
;show methods
(clear)
(% (property set1) (.1 .2))
(% (property set2) (.5 .54))
(% (property set3) (.1 .53))
(% (property set4) (.1 .6))
(subs set2 (set1))
(mem object (set1 set2 set3 set4))
(set trace on)
(set method 1)
(prob (property object))
(set method 2)
(prob (property object))
(set method 3)
(prob (property object))
(set method 4)
(prob (property object))
(set trace off)
(set method 1)
(prob (property object))
(set method 2)
(prob (property object))
(set method 3)
(prob (property object))
(set method 4)
(prob (property object))
```

Sample Session

A sample session using Sun Common Lisp and the sample data file:

```
> (load "mkuis")
;;; Loading binary file "mkuis.sbin"
;;; Loading binary file "definitions.sbin"
;;; Loading binary file "io.sbin"
;;; Loading binary file "math.sbin"
;;; Loading binary file "front-end.sbin"
;;; Loading binary file "process.sbin"
;;; Loading binary file "sets.sbin"
;;; Loading binary file "select.sbin"
;;; Loading binary file "methods.sbin"
;;; Loading binary file "genss.sbin"
;;; Loading binary file "compclass.sbin"
;;; Loading binary file "cover.sbin"
;;; Loading binary file "xp.sbin"

Mucit's Kyburgian Uncertain Inference Shell
Common Lisp Version 0.40  August 13, 1990


shell is reset

INF-SHELL==> (load "example")
LOADING ==>(RESET)

shell is reset
LOADING ==>(SET TIMING OFF)
LOADING ==>(S% (FLY BIRD) (100000 90000))
@ 0.95confidence level % (FLY BIRD) in [0.89817 , 0.90181]

LOADING ==>(S% (FLY PENGUIN) (1000 1))
@ 0.95confidence level % (FLY PENGUIN) in [0.00018 , 0.00548]

LOADING ==>(SUBS PENGUIN (BIRD))
LOADING ==>(MEM TWEETY (BIRD))
LOADING ==>(PROB (FLY TWEETY))

Processing query :(PROB (FLY TWEETY))  Using Method 4
All supersets:
TWEETY
BIRD
```

```
All non-trivial candidates :
<BIRD , [0.89817 , 0.90181]>

And the reference class is :
<BIRD , [0.89817 , 0.90181]>
LOADING ==>(MEM TWEETY (PENGUIN))
LOADING ==>(PROB (FLY TWEETY))

Processing query :(PROB (FLY TWEETY))  Using Method 4
All supersets:
TWEETY
PENGUIN
BIRD


All non-trivial candidates :
<PENGUIN , [0.00018 , 0.00548]>
<BIRD , [0.89817 , 0.90181]>

conflict between
<PENGUIN , [0.00018 , 0.00548]>
<BIRD , [0.89817 , 0.90181]>

Winner by subset:
<PENGUIN , [0.00018 , 0.00548]>

Will combine the following
<PENGUIN , [0.00018 , 0.00548]>

And the reference class is :
<PENGUIN , [0.00018 , 0.00548]>
LOADING ==>(SET CONFIDENCE 0.99)
LOADING ==>(PROB (FLY TWEETY))

Processing query :(PROB (FLY TWEETY))  Using Method 4
All supersets:
TWEETY
PENGUIN
BIRD


All non-trivial candidates :
```

```
<PENGUIN , [0.00012 , 0.00840]>
<BIRD , [0.89754 , 0.90241]>

conflict between
<PENGUIN , [0.00012 , 0.00840]>
<BIRD , [0.89754 , 0.90241]>

Winner by subset:
<PENGUIN , [0.00012 , 0.00840]>

Will combine the following
<PENGUIN , [0.00012 , 0.00840]>

And the reference class is :
<PENGUIN , [0.00012 , 0.00840]>
LOADING ==>(SET CONFIDENCE 0.51)
LOADING ==>(PROB (FLY TWEETY))

Processing query :(PROB (FLY TWEETY))  Using Method 4
All supersets:
TWEETY
PENGUIN
BIRD


All non-trivial candidates :
<PENGUIN , [0.00060 , 0.00167]>
<BIRD , [0.89950 , 0.90049]>

conflict between
<PENGUIN , [0.00060 , 0.00167]>
<BIRD , [0.89950 , 0.90049]>

Winner by subset:
<PENGUIN , [0.00060 , 0.00167]>

Will combine the following
<PENGUIN , [0.00060 , 0.00167]>

And the reference class is :
<PENGUIN , [0.00060 , 0.00167]>
LOADING ==>(SET TRACE OFF)

Processing query :(PROB (FLY TWEETY))  Using Method 4
```

```
And the reference class is :
<PENGUIN , [0.00060 , 0.00167]>

Processing query :(PROB (FLY TWEETY))  Using Method 4
And the reference class is :
<PENGUIN , [0.00060 , 0.00167]>
Elapsed Real Time = 0.05 seconds
Total Run Time    = 0.06 seconds
User Run Time     = 0.05 seconds
System Run Time   = 0.01 seconds
Process Page Faults = 0
Dynamic Bytes Consed = 0

shell is reset
LOADING ==>(SET TIMING OFF)
LOADING ==>(% (P A) (0.1 0.2))
LOADING ==>(% (P B) (0.3 0.4))
LOADING ==>(% (P C) (0.15 0.45))
LOADING ==>(SUBS C (A))
LOADING ==>(MEM X (A B C))
LOADING ==>(PROB (P X))

Processing query :(PROB (P X))  Using Method 4
All supersets:
X
B
C
A


All non-trivial candidates :
<B , [0.30000 , 0.40000]>
<C , [0.15000 , 0.45000]>
<A , [0.10000 , 0.20000]>

conflict between
<B , [0.30000 , 0.40000]>
<A , [0.10000 , 0.20000]>

No reflection

conflict between
<C , [0.15000 , 0.45000]>
<A , [0.10000 , 0.20000]>
```

```
Winner by subset:
<C , [0.15000 , 0.45000]>

Will combine the following
<C , [0.15000 , 0.45000]>

And the reference class is :
<C , [0.15000 , 0.45000]>
LOADING ==>(CLEAR)
LOADING ==>(% (P A) (0.1 0.2))
LOADING ==>(% (P B) (0.3 0.4))
LOADING ==>(% (P C) (0.35 0.45))
LOADING ==>(SUBS C (B))
LOADING ==>(MEM X (A B C))
LOADING ==>(PROB (P X))

Processing query :(PROB (P X))  Using Method 4
All supersets:
X
A
C
B


All non-trivial candidates :
<A , [0.10000 , 0.20000]>
<C , [0.35000 , 0.45000]>
<B , [0.30000 , 0.40000]>

conflict between
<A , [0.10000 , 0.20000]>
<C , [0.35000 , 0.45000]>

No reflection

conflict between
<A , [0.10000 , 0.20000]>
<B , [0.30000 , 0.40000]>

No reflection

conflict between
<C , [0.35000 , 0.45000]>
```

```
<B , [0.30000 , 0.40000]>

Winner by subset:
<C , [0.35000 , 0.45000]>

Will combine the following
<C , [0.35000 , 0.45000]>
<A , [0.10000 , 0.20000]>

And the reference class is :
<(OR A C) , [0.10000 , 0.45000]>
LOADING ==>(CLEAR)
LOADING ==>(% (P A) (0.1 0.2))
LOADING ==>(% (P B) (0.3 0.4))
LOADING ==>(% (P (I A B)) (0.35 0.45))
LOADING ==>(MEM X (A B))
LOADING ==>(PROB (P X))

Processing query :(PROB (P X))  Using Method 4
All supersets:
X
(I A B)
A
B


All non-trivial candidates :
<(I A B) , [0.35000 , 0.45000]>
<A , [0.10000 , 0.20000]>
<B , [0.30000 , 0.40000]>

conflict between
<(I A B) , [0.35000 , 0.45000]>
<A , [0.10000 , 0.20000]>

Winner by subset:
<(I A B) , [0.35000 , 0.45000]>

conflict between
<(I A B) , [0.35000 , 0.45000]>
<B , [0.30000 , 0.40000]>

Winner by subset:
<(I A B) , [0.35000 , 0.45000]>
```

```
conflict between
<A , [0.10000 , 0.20000]>
<B , [0.30000 , 0.40000]>

No reflection

Will combine the following
<(I A B) , [0.35000 , 0.45000]>

And the reference class is :
<(I A B) , [0.35000 , 0.45000]>
LOADING ==>(CLEAR)
LOADING ==>(% (PROPERTY SET1) (0.1 0.2))
LOADING ==>(% (PROPERTY SET2) (0.5 0.54))
LOADING ==>(% (PROPERTY SET3) (0.1 0.53))
LOADING ==>(% (PROPERTY SET4) (0.1 0.6))
LOADING ==>(SUBS SET2 (SET1))
LOADING ==>(MEM OBJECT (SET1 SET2 SET3 SET4))
LOADING ==>(SET TRACE ON)
LOADING ==>(SET METHOD 1)
LOADING ==>(PROB (PROPERTY OBJECT))

Processing query :(PROB (PROPERTY OBJECT))  Using Method 1
All supersets:
OBJECT
SET2
SET1
SET3
SET4


All non-trivial candidates :
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>
<SET3 , [0.10000 , 0.53000]>
<SET4 , [0.10000 , 0.60000]>

4 is to combine
Creating 2 place XPs
5XP inference structures in this iteration
4 is to combine
Creating 3 place XPs
2XP inference structures in this iteration
```

```
XP generation finished.  A total of 7 inference structures generated
All candidates
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>
<(XP SET1 SET3) , [0.01220 , 0.21992]>
<(XP SET1 SET4) , [0.01220 , 0.27273]>
<(XP SET4 SET3 SET1) , [0.00137 , 0.29720]>
<SET3 , [0.10000 , 0.53000]>
<(XP SET2 SET3) , [0.10000 , 0.56967]>
<SET4 , [0.10000 , 0.60000]>
<(XP SET2 SET4) , [0.10000 , 0.63780]>
<(XP SET3 SET4) , [0.01220 , 0.62846]>
<(XP SET4 SET3 SET2) , [0.01220 , 0.66507]>

Considering :
<SET2 , [0.50000 , 0.54000]>

Disagreement with:
<SET1 , [0.10000 , 0.20000]>

Candidate survived
Disagreement with:
<(XP SET1 SET3) , [0.01220 , 0.21992]>

Doesn't dominate

Considering :
<(XP SET1 SET3) , [0.01220 , 0.21992]>

Disagreement with:
<SET2 , [0.50000 , 0.54000]>

Doesn't dominate

Considering :
<(XP SET1 SET4) , [0.01220 , 0.27273]>

Disagreement with:
<SET2 , [0.50000 , 0.54000]>

Doesn't dominate

Considering :
<(XP SET4 SET3 SET1) , [0.00137 , 0.29720]>
```

```
Disagreement with:
<SET2 , [0.50000 , 0.54000]>

Doesn't dominate

Considering :
<SET3 , [0.10000 , 0.53000]>

Disagreement with:
<SET2 , [0.50000 , 0.54000]>

Doesn't dominate

Considering :
<(XP SET2 SET3) , [0.10000 , 0.56967]>

Disagreement with:
<(XP SET1 SET3) , [0.01220 , 0.21992]>

Candidate survived
Disagreement with:
<(XP SET1 SET4) , [0.01220 , 0.27273]>

Doesn't dominate

Considering :
<SET4 , [0.10000 , 0.60000]>

Disagreement with:
<(XP SET1 SET3) , [0.01220 , 0.21992]>

Doesn't dominate

Considering :
<(XP SET2 SET4) , [0.10000 , 0.63780]>

Disagreement with:
<(XP SET1 SET3) , [0.01220 , 0.21992]>

Doesn't dominate

Considering :
<(XP SET3 SET4) , [0.01220 , 0.62846]>
```

```
Disagreement with:
<(XP SET4 SET3 SET1) , [0.00137 , 0.29720]>

Doesn't dominate

Considering :
<(XP SET4 SET3 SET2) , [0.01220 , 0.66507]>

Disagreement with:
<(XP SET4 SET3 SET1) , [0.00137 , 0.29720]>

Candidate survived
And the reference class is :
<(XP SET4 SET3 SET2) , [0.01220 , 0.66507]>
LOADING ==>(SET METHOD 2)
LOADING ==>(PROB (PROPERTY OBJECT))

Processing query :(PROB (PROPERTY OBJECT))  Using Method 2
All supersets:
OBJECT
SET2
SET1
SET3
SET4


All non-trivial candidates :
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>
<SET3 , [0.10000 , 0.53000]>
<SET4 , [0.10000 , 0.60000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>

Winner by subset:
<SET2 , [0.50000 , 0.54000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET3 , [0.10000 , 0.53000]>
```

```
No reflection

After conflicts are resolved
<SET4 , [0.10000 , 0.60000]>
<SET3 , [0.10000 , 0.53000]>
<SET2 , [0.50000 , 0.54000]>

3 is to combine
Creating 2 place XPs
3XP inference structures in this iteration
3 is to combine
Creating 3 place XPs
1XP inference structures in this iteration
XP generation finished.  A total of 4 inference structures generated
All candidates with XP's
<SET2 , [0.50000 , 0.54000]>
<SET3 , [0.10000 , 0.53000]>
<(XP SET3 SET2) , [0.10000 , 0.56967]>
<SET4 , [0.10000 , 0.60000]>
<(XP SET4 SET2) , [0.10000 , 0.63780]>
<(XP SET4 SET3) , [0.01220 , 0.62846]>
<(XP SET2 SET3 SET4) , [0.01220 , 0.66507]>

Considering :
<SET2 , [0.50000 , 0.54000]>

Disagreement with:
<SET3 , [0.10000 , 0.53000]>

Doesn't dominate

Considering :
<SET3 , [0.10000 , 0.53000]>

Disagreement with:
<SET2 , [0.50000 , 0.54000]>

Doesn't dominate

Considering :
<(XP SET3 SET2) , [0.10000 , 0.56967]>

And the reference class is :
<(XP SET3 SET2) , [0.10000 , 0.56967]>
```

```
LOADING ==>(SET METHOD 3)
LOADING ==>(PROB (PROPERTY OBJECT))

Processing query :(PROB (PROPERTY OBJECT))  Using Method 3
All supersets:
OBJECT
SET2
SET1
SET3
SET4


All non-trivial candidates :
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>
<SET3 , [0.10000 , 0.53000]>
<SET4 , [0.10000 , 0.60000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>

Winner by subset:
<SET2 , [0.50000 , 0.54000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET3 , [0.10000 , 0.53000]>

No reflection

After conflicts are resolved
<SET4 , [0.10000 , 0.60000]>
<SET3 , [0.10000 , 0.53000]>
<SET2 , [0.50000 , 0.54000]>

Will combine the following
<SET2 , [0.50000 , 0.54000]>
<SET3 , [0.10000 , 0.53000]>

2 inference structures to combine
Creating 2 place XPs
1XP inference structures in this iteration
XP generation finished.  A total of 1 inference structures generated
```

```
Considering :
<(XP SET2 SET3) , [0.10000 , 0.56967]>

And the reference class is :
<(XP SET2 SET3) , [0.10000 , 0.56967]>
LOADING ==>(SET METHOD 4)
LOADING ==>(PROB (PROPERTY OBJECT))

Processing query :(PROB (PROPERTY OBJECT))  Using Method 4
All supersets:
OBJECT
SET2
SET1
SET3
SET4


All non-trivial candidates :
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>
<SET3 , [0.10000 , 0.53000]>
<SET4 , [0.10000 , 0.60000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>

Winner by subset:
<SET2 , [0.50000 , 0.54000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET3 , [0.10000 , 0.53000]>

No reflection

Will combine the following
<SET4 , [0.10000 , 0.60000]>
<SET3 , [0.10000 , 0.53000]>
<SET2 , [0.50000 , 0.54000]>

And the reference class is :
<(OR SET2 SET3) , [0.10000 , 0.54000]>
LOADING ==>(SET TRACE OFF)
```

```
Processing query :(PROB (PROPERTY OBJECT))  Using Method 1
4 is to combine
Creating 2 place XPs
5XP inference structures in this iteration
4 is to combine
Creating 3 place XPs
2XP inference structures in this iteration
XP generation finished.  A total of 7 inference structures generated
And the reference class is :
<(XP SET4 SET3 SET2) , [0.01220 , 0.66507]>

Processing query :(PROB (PROPERTY OBJECT))  Using Method 2
3 is to combine
Creating 2 place XPs
3XP inference structures in this iteration
3 is to combine
Creating 3 place XPs
1XP inference structures in this iteration
XP generation finished.  A total of 4 inference structures generated
And the reference class is :
<(XP SET3 SET2) , [0.10000 , 0.56967]>

Processing query :(PROB (PROPERTY OBJECT))  Using Method 3
2 inference structures to combine
Creating 2 place XPs
1XP inference structures in this iteration
XP generation finished.  A total of 1 inference structures generated
And the reference class is :
<(XP SET2 SET3) , [0.10000 , 0.56967]>

Processing query :(PROB (PROPERTY OBJECT))  Using Method 4
And the reference class is :
<(OR SET2 SET3) , [0.10000 , 0.54000]>
eof reached

INF-SHELL==> (bye)
Bye.
#P"/home/castor/u6/mucit/statinf/mkuis.sbin"
>
```

References

[1] Henry E. Kyburg. The reference class. *Philosophy of Science*, (50), 1983.